paddy

May 30, 2025

1 Assigment 3

1.1 Task 1 - Paddy Disease Detection

As stated in the assignment, we will create a model to detect paddy diseases using the dataset provided. I started by looking and implementing the code that Jeremy Howard provided, and managed to get similar results to the ones he got, although the provided compute is not suitable for his models and resulted in a very long training time. I therefore started to play with the model and tried to improve the results while keeping the training time at tolerable level.

I initially did a test run of what Jeremy Howard had done, just to see if I could replicate his results. After several hours of training on a Google Colab notebook, which offers 16 GB of VRAM compared to our 8 GB at the computer lab, I got a similar result, but was left with no more compute tokens. I therefore realised that in order to continue working on this task with limited compute I had to make some changes.

The first thing I did was to try different architectures. I started with the Resnet50 architecture, which is a very popular, but now an old architecture for image classification tasks, and to reduce the training time, I used a smaller input size of 224x224. The results were not very good, with a validation accuracy of around 0.8. I then tried the Resnet101 architecture, which is a deeper version of Resnet50, and the results improved slightly, with a validation accuracy of around 0.86. I could have tried to improve these resnet models more, with more augmentations and fine-tuning the hyperparameters, but I knew that Mr. Howard has used different architectures, such as ConvNext and ViT.

I therefore tried the ConvNext Small 22k and found that it performed much better than a Resnet101 and similar architectures. This is likely due to the fact that ConvNext is a modernized convolutional neural network inspired by the transformer era (like ViT), while it keeps the efficiency of CNNs and that the Convnext model is pretrained on ImageNet-22K (22,000 classes), and Resnet is usually pretrained on ImageNet-1K (1,000 classes). The ConvNext has a tendency to build stronger hierarchical features due to its modern block structure and larger receptive fields, and since the disease is not very visible, the model needs to learn more complex features.

I then tried the ConvNext Small 22k and found that it performed much better than a Resnet101 or similar architectures. This is likely due to the fact that ConvNext is a modernized convolutional neural network inspired by the transformer era (like ViT), while it keeps the efficiency of CNNs and that the Convnext model is pretrained on ImageNet-22K (22,000 classes), and Resnet is usually pretrained on ImageNet-1K (1,000 classes). The ConvNext has a tendency to build stronger hierarchical features due to its modern block structure and larger receptive fields, and since the disease is not very visible, the model needs to learn more complex features.

To try to make the model even better I created an ensemble of Resnet models and ConvNext models with different hyperparameters and augmentations. The results were very good, with a score of around 0.97846 on Kaggle, while still keeping the training time to a reasonable level (34 min, 8 GB GPU memory). The first ensemble looked like this:

Arch	Image size	Epochs	Augmentations	Error rate	Training time (approx)
resnet50	224	10	scale (min 75%)	0.144	7 min
resnet101	224	10	scale (min 75%)	0.142	10 min
convnext_small_in22k	224	10	scale (min 75%)	0.028	11 min
vit_small_patch16_224	224	10	scale (min 75%)	0.026	6 min

Kaggle score	0.97846

The ensemble works similarly to Jeremy's model, where the predictions of each model are averaged to get the final prediction.

After realizing that the resnets were not performing as well as the ConvNext and ViT models, I decided to remove them from the ensemble and only keep the ConvNext and ViT models. The next ensemble turned out like this:

Arch	Image size	Epochs	Augmentations	Error rate	Training time (approx)
convnext_small_in22k	224	10	scale (min 75%)	0.024	11 min
vit_small_patch16_224	224	10	scale (min 75%)	0.029	6 min

Kaggle score	0.98306

Which resulted in a score of 0.98306 on Kaggle, which is slightly better. An ensemble with only two models is not really enough when the descision is based on the average of the predictions though, so I added two more variants of the ConvNext model and the ViT model with different augmentations and hyperparameters. The improved ensemble looked like this:

Arch	Image size	Epochs	Augmentations	Error rate	Training time (approx)
convnext_small_in22k	299	10	scale (min 75%)	0.024	16 min
convnext_small_in22k	224	10	scale (min 75%)	0.024	11 min
vit_small_patch16_224	224	10	scale (min 75%)	0.029	6 min

Kaggle score	0.98385

This resulted in a score of 0.98385 on Kaggle. The training time was around 30 minutes, and the GPU memory usage was around 8 GB.

As a final test, I added a nother ConvNext model with a different input size of 320. Which resulted in a slightly higher score (0.98539) on Kaggle, but with the downside of a substantially longer training time of around 1 hour and 34 minutes. The final ensemble looked like this:

Arch	Image size	Epochs	Augmentations	Error rate	Training time (approx)
convnext_small_in22k	320	10	scale (min 75%)	0.021	60 min
convnext_small_in22k	299	10	scale (min 75%)	0.024	16 min
convnext_small_in22k	224	10	scale (min 75%)	0.025	11 min
vit_small_patch16_224	244	10	scale (min 75%)	0.032	6 min

Kaggle score	0.98539

I spent a lot of time trying out if I could improve the GPU/CPU usage by rescaling the whole dataset to the desired sizes before starting any training, without much success. The resize function and scale augmentations in Fast AI are not the bottle neck of the training.

In terms of other augmentations, I used Fast AI's standard augmentations which is defined as follows:

Augmentation	Value/Setting
Random horizontal flips	Yes
Maximum degree of rotation	10°
Minimum zoom	1.0

Augmentation	Value/Setting
Maximum zoom	1.1
Maximum scale for changing brightness	0.2
Maximum value for changing warp	0.2
Probability of affine transformation	0.2
Probability of brightness/contrast	0.75
Padding mode	Reflection

We can also see from the confusion matricies that the models perform very well regardless of the class imbalances in the training set. Class imbalance can be a major issue when training, due to the models prioritsing the classes with the most samples. If we had a dataset with 99% of one class, the model would in many cases disregard the other classes due to the small quantity. There are many ways to solve this though, where collecting more data, syntesising more data, changing the performance metric and penelising the model are possible options. This is not tested in this assignment, but should be included for future work.

1.2 Final notebook of Paddy Disease Detection

[ ]:
```
!pip install fastkaggle fastai
```

[ ]:
```
from fastkaggle import *
from fastai.vision.all import *


comp = 'paddy-disease-classification'

path = setup_comp(comp, install='fastai "timm>=0.6.2.dev0"')
path.ls()

trn_path = path / "train_images"
tst_path = path / "test_images"
```

1.2.1 Training function

[ ]:
```
def train(arch, size, item=Resize(480, method='squish'), accum=1,␣
 ↪finetune=True, epochs=12):
    dls = ImageDataLoaders.from_folder(trn_path, valid_pct=0.2, item_tfms=item,
        batch_tfms=aug_transforms(size=size, min_scale=0.75), bs=64//accum)
    cbs = GradientAccumulation(64) if accum else []
    learn = vision_learner(dls, arch, metrics=error_rate, cbs=cbs).to_fp16()
    if finetune:
        learn.fine_tune(epochs, 0.01)
        learn.validate()
        tst_files = get_image_files(tst_path)

        interp = ClassificationInterpretation.from_learner(learn)
```

```
        interp.plot_confusion_matrix()
        interp.plot_top_losses(9)

        test_dl = learn.dls.test_dl(tst_files)
        return learn.tta(dl=test_dl)
    else:
        learn.unfreeze()
        learn.fit_one_cycle(epochs, 0.01)
```

1.2.2 Model architecture

[ ]:
```
res = 512,512
models = {
    'convnext_small_in22k': {
        (Resize(res), 224),
        (Resize(res), 299),
        (Resize(res), 320),
    }, 'vit_small_patch16_224': {
         (Resize(res), 224),
    }
}
```

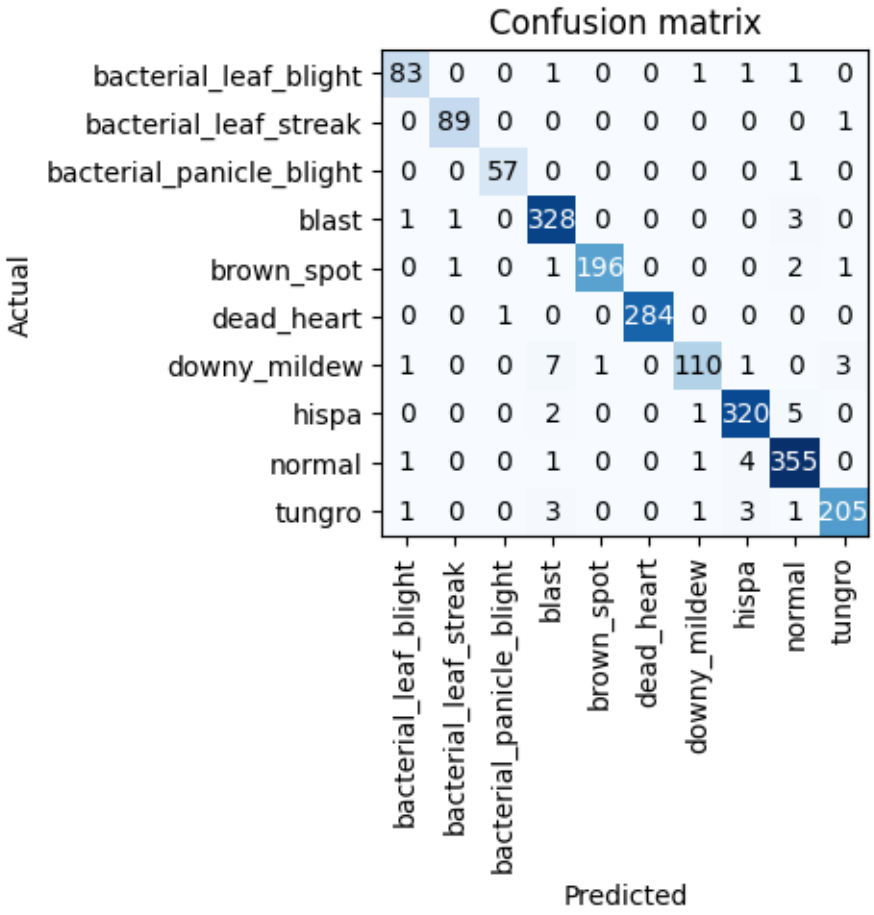
[ ]:
```
import gc
tta_res = []

for arch,details in models.items():
    for item,size in details:
        print('---',arch)
        print(size)
        print(item.name)
        tta_res.append(train(arch, size, item=item, accum=2, epochs=10)) #,␣
 ↪epochs=1))
        gc.collect()
        torch.cuda.empty_cache()
```

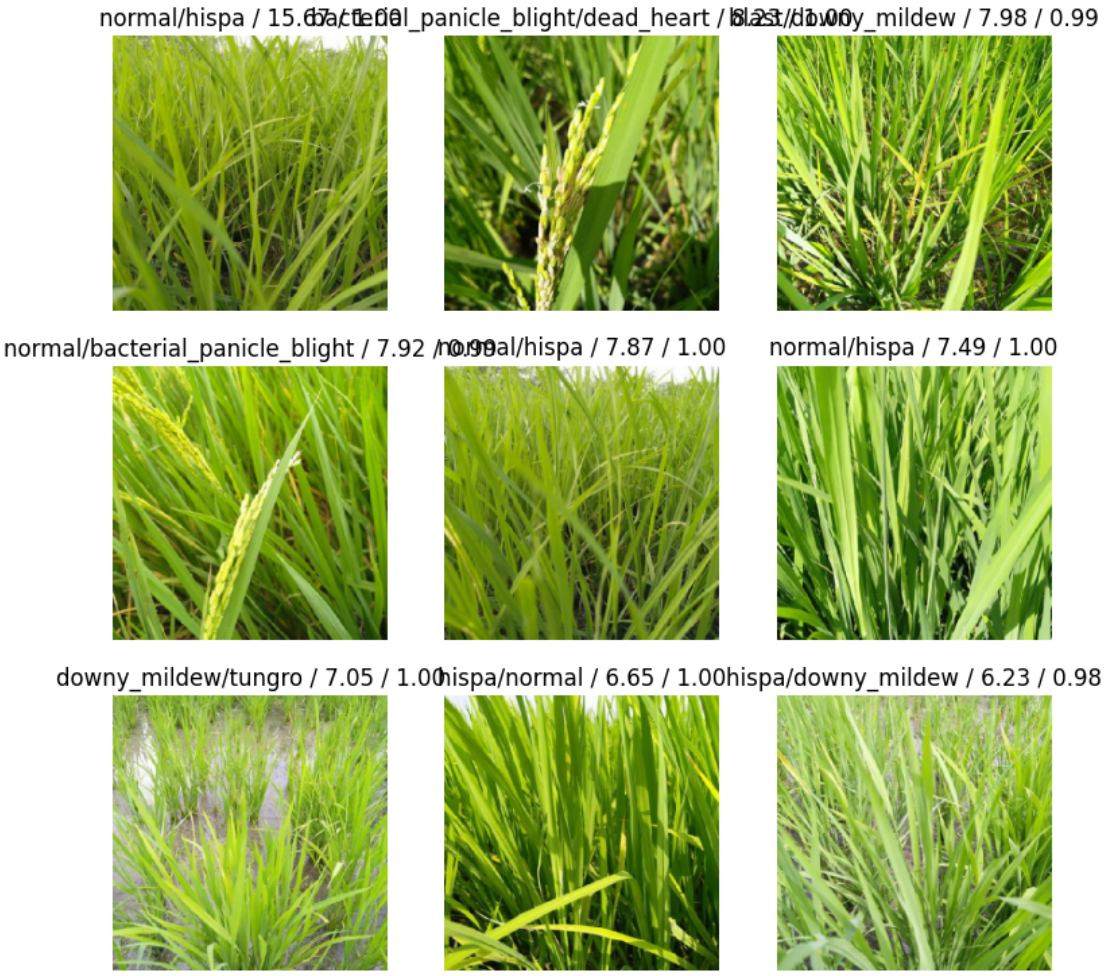
1.2.3 Output

— convnext_small_in22k 224 Resize – {'size': (512, 512), 'method': 'crop', 'pad_mode': 'reflection', 'resamples': (<Resampling.BILINEAR: 2>, <Resampling.NEAREST: 0>), 'p': 1.0}

epoch	train_loss	valid_loss	error_rate	time
0	0.522924	0.291351	0.092744	01:07
1	0.474965	0.308874	0.100913	01:04
2	0.409522	0.312514	0.094185	01:05
3	0.247534	0.196876	0.058626	01:04
4	0.228018	0.183603	0.054301	01:04
5	0.169902	0.132934	0.041326	01:05

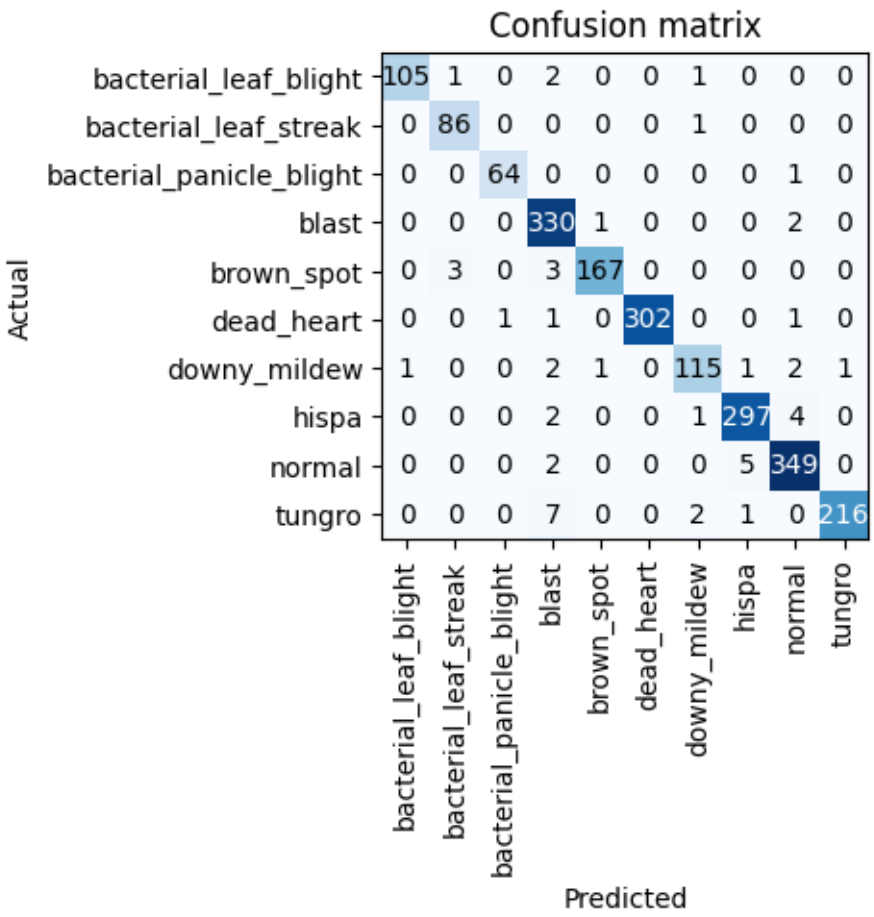
epoch	train_loss	valid_loss	error_rate	time
6	0.137842	0.120899	0.034599	01:05
7	0.091253	0.113226	0.027871	01:05
8	0.066507	0.107186	0.027871	01:04
9	0.059744	0.105628	0.025949	01:03

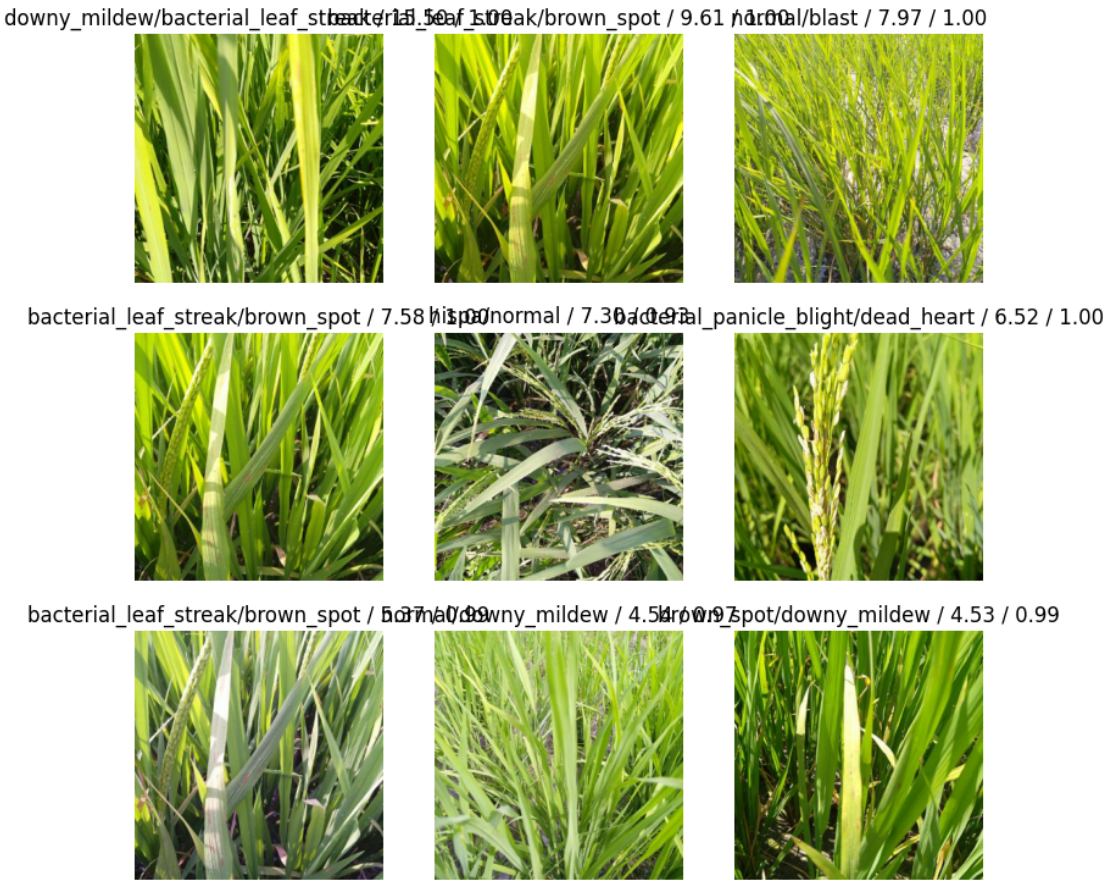




— convnext_small_in22k 299 Resize – {'size': (512, 512), 'method': 'crop', 'pad_mode': 'reflection', 'resamples': (<Resampling.BILINEAR: 2>, <Resampling.NEAREST: 0>), 'p': 1.0}

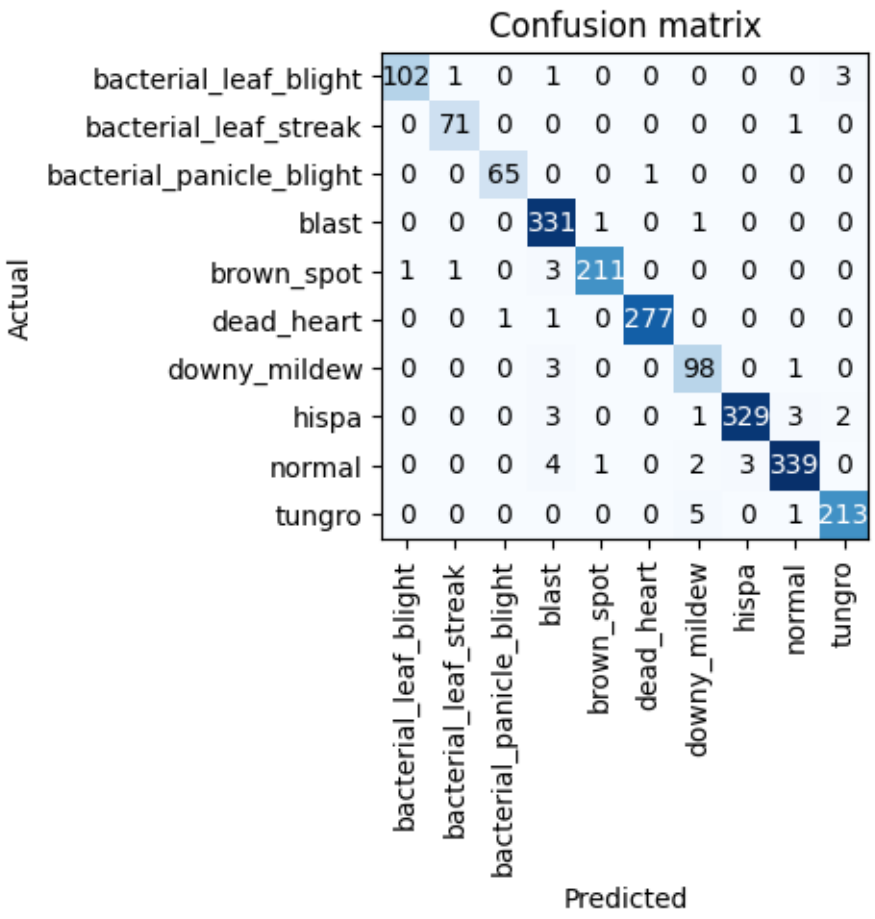
epoch	train_loss	valid_loss	error_rate	time
0	0.482487	0.301711	0.093224	01:33
1	0.404623	0.259028	0.085055	01:31
2	0.334706	0.276585	0.089861	01:31
3	0.293459	0.220637	0.063431	01:32
4	0.209628	0.169217	0.054301	01:36
5	0.145868	0.137486	0.039404	01:37
6	0.117963	0.117456	0.033157	01:36
7	0.088826	0.099379	0.025949	01:35
8	0.081176	0.086329	0.024027	01:37
9	0.068929	0.085969	0.024027	01:35


Confusion matrix
bacterial_leaf_blight 105 1 0 2 0 0 1 0 0 0
bacterial_leaf_streak 0 86 0 0 0 0 1 0 0 0
bacterial_panicle_blight 0 0 64 0 0 0 0 0 1 0
blast 0 0 0 330 1 0 0 0 2 0
brown_spot 0 3 0 3 167 0 0 0 0 0
dead_heart 0 0 1 1 0 302 0 0 1 0
downy_mildew 1 0 0 2 1 0 115 1 2 1
hispa 0 0 0 2 0 0 1 297 4 0
normal 0 0 0 2 0 0 0 5 349 0
tungro 0 0 0 7 0 0 2 1 0 216
Actual
bacterial_leaf_blight
bacterial_leaf_streak
bacterial_panicle_blight
blast
brown_spot
dead_heart
downy_mildew
hispa
normal
tungro
Predicted



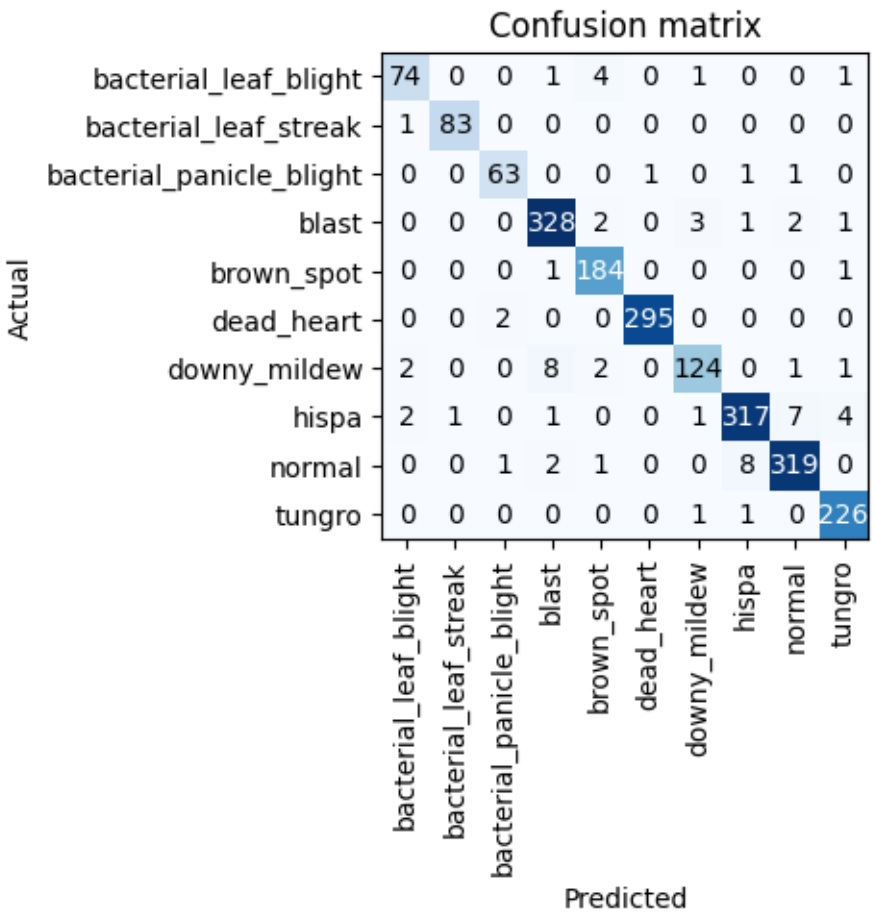
— convnext_small_in22k 320 Resize – {'size': (512, 512), 'method': 'crop', 'pad_mode': 'reflection', 'resamples': (<Resampling.BILINEAR: 2>, <Resampling.NEAREST: 0>), 'p': 1.0}

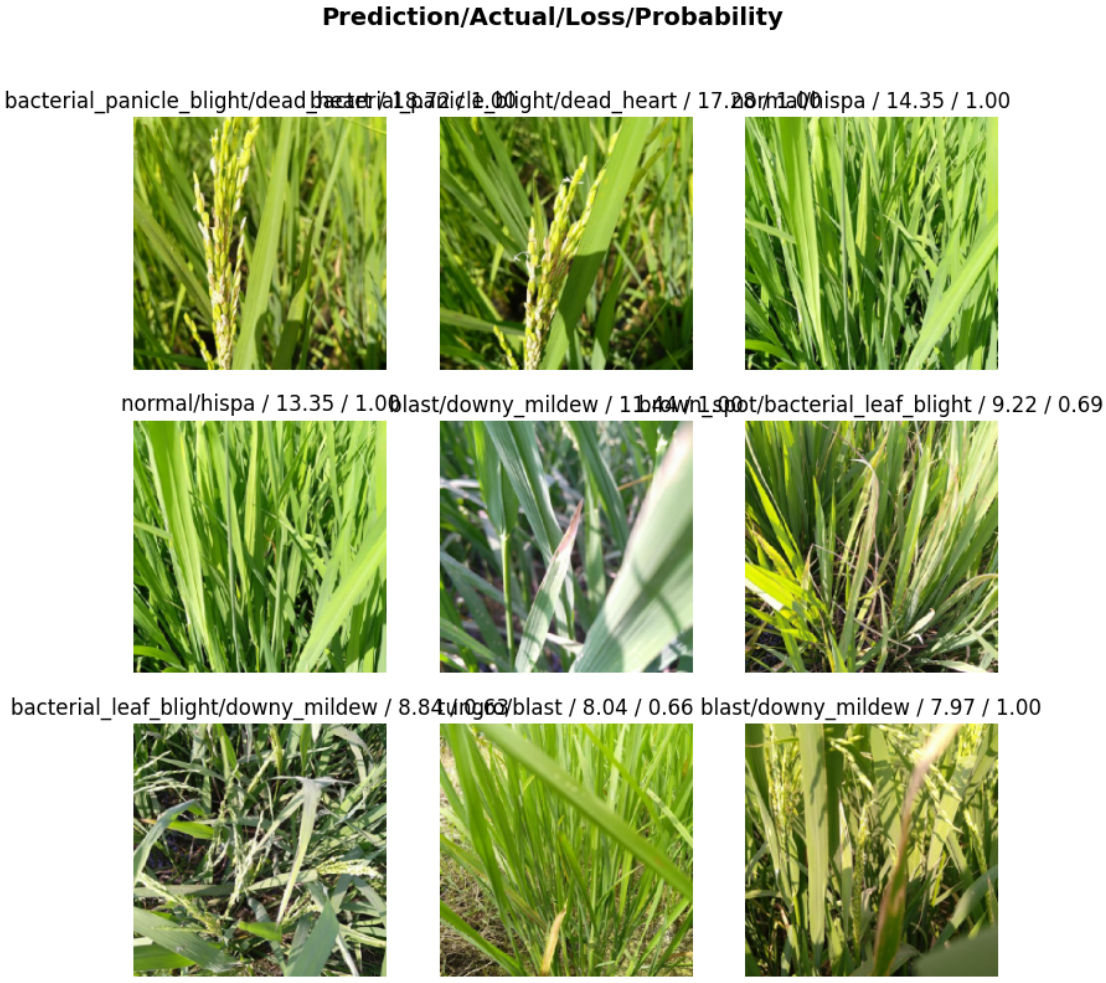
epoch	train_loss	valid_loss	error_rate	time
0	0.475270	0.282908	0.095627	05:25
1	0.394569	0.250861	0.079769	05:33
2	0.362648	0.222830	0.066314	05:53
3	0.320339	0.183621	0.059106	05:57
4	0.201428	0.160160	0.044690	05:56
5	0.150571	0.149583	0.039404	05:56
6	0.119630	0.112060	0.031235	05:56
7	0.097585	0.100967	0.022105	05:53
8	0.061469	0.091153	0.023546	05:44
9	0.062056	0.088169	0.021624	05:44


Confusion matrix
bacterial_leaf_blight 102 1 0 1 0 0 0 0 0 3
bacterial_leaf_streak 0 71 0 0 0 0 0 0 1 0
bacterial_panicle_blight 0 0 65 0 0 1 0 0 0 0
blast 0 0 0 331 1 0 1 0 0 0
brown_spot 1 1 0 3 211 0 0 0 0 0
dead_heart 0 0 1 1 0 277 0 0 0 0
downy_mildew 0 0 0 3 0 0 98 0 1 0
hispa 0 0 0 3 0 0 1 329 3 2
normal 0 0 0 4 1 0 2 3 339 0
tungro 0 0 0 0 0 0 5 0 1 213
Actual
bacterial_leaf_blight
bacterial_leaf_streak
bacterial_panicle_blight
blast
brown_spot
dead_heart
downy_mildew
hispa
normal
tungro
Predicted

— vit_small_patch16_224 224 Resize – {'size': (512, 512), 'method': 'crop', 'pad_mode': 'reflection', 'resamples': (<Resampling.BILINEAR: 2>, <Resampling.NEAREST: 0>), 'p': 1.0}

epoch	train_loss	valid_loss	error_rate	time
0	0.629896	0.369405	0.114368	00:34
1	0.473076	0.325375	0.100432	00:34
2	0.472268	0.353715	0.104277	00:34
3	0.342373	0.235656	0.069197	00:35
4	0.280413	0.292275	0.080730	00:35
5	0.225597	0.211669	0.057184	00:35
6	0.145305	0.163454	0.038924	00:35
7	0.135815	0.162330	0.038443	00:35
8	0.086223	0.154728	0.032196	00:35
9	0.091272	0.152397	0.032677	00:34


Confusion matrix
bacterial_leaf_blight 74 0 0 1 4 0 1 0 0 1
bacterial_leaf_streak 1 83 0 0 0 0 0 0 0 0
bacterial_panicle_blight 0 0 63 0 0 1 0 1 1 0
blast 0 0 0 328 2 0 3 1 2 1
brown_spot 0 0 0 1 184 0 0 0 0 1
dead_heart 0 0 2 0 0 295 0 0 0 0
downy_mildew 2 0 0 8 2 0 124 0 1 1
hispa 2 1 0 1 0 0 1 317 7 4
normal 0 0 1 2 1 0 0 8 319 0
tungro 0 0 0 0 0 0 1 1 0 226
Actual
bacterial_leaf_blight
bacterial_leaf_streak
bacterial_panicle_blight
blast
brown_spot
dead_heart
downy_mildew
hispa
normal
tungro
Predicted



[ ]:
```
save_pickle('tta_paddy.pkl', tta_res)
```

[ ]:
```
tta_prs = first(zip(*tta_res))
avg_pr = torch.stack(tta_prs).mean(0)
dls = ImageDataLoaders.from_folder(trn_path, valid_pct=0.2,␣
 ↪item_tfms=Resize(480, method='squish'),
    batch_tfms=aug_transforms(size=224, min_scale=0.75))

tst_files = get_image_files(tst_path)

idxs = avg_pr.argmax(dim=1)
vocab = np.array(dls.vocab)

submission = pd.DataFrame({
    "image_id": [f.name for f in tst_files.items],
    "label": vocab[idxs]
})
submission.to_csv("paddy-submission.csv", index=False)
```

[ ]:
```
if not iskaggle:
    from kaggle import api
    api.competition_submit_cli('paddy-submission.csv', 'ensemble 512px 10e',␣
 ↪comp)
```